

TURING

图灵程序设计丛书

[PACKT]
PUBLISHING

Introduction to JVM Languages

Java虚拟机基础教程

【荷】Vincent van der Leun 著

袁国忠 译

- 运用大量示例探讨Java、Scala、Clojure、Kotlin和Groovy的核心概念

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS



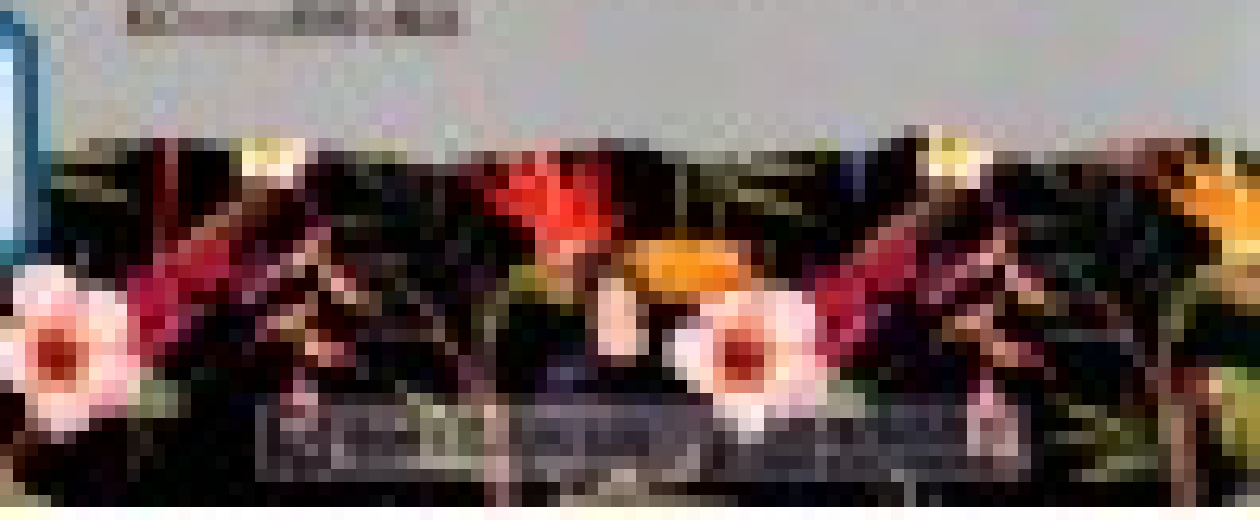
Introduction to JVM Language

Java虚拟机基础教程

◎ 清华大学出版社

◎ 2009年

■ 清华大学出版社
地址：北京清华大学学研大厦A座
邮编：100084
电话：(010) 62770175
http://www.tup.tsinghua.edu.cn



TURING

图灵程序设计

Introduction to JVM Languages

Java虚拟机基础教程

【荷】 Vincent van der Leun 著
袁国忠 译

人民邮电出版社
北 京

图书在版编目 (C I P) 数据

Java虚拟机基础教程 / (荷) 文森特·范德利昂著 ;
袁国忠译. — 北京 : 人民邮电出版社, 2018. 2
(图灵程序设计丛书)
ISBN 978-7-115-47779-8

I. ①J… II. ①文… ②袁… III. ①JAVA语言—程序
设计 IV. ①TP312.8

中国版本图书馆CIP数据核字(2018)第010917号

内 容 提 要

本书概述 Java 虚拟机 (JVM) 及其特性, 并用大量示例详细介绍了 Java、Scala、Clojure、Kotlin 和 Groovy 这 5 种基于 JVM 的语言。具体而言, 首先概述了 Java 平台, 紧接着详细阐述了 JVM, 然后分别介绍了上述各种语言的基础知识和核心概念, 并运用它们开发项目、创建应用程序。

本书适合所有 Java 开发人员以及对 JVM 感兴趣的读者。

-
- ◆ 著 [荷] Vincent van der Leun
译 袁国忠
责任编辑 岳新欣
执行编辑 李 敏
责任印制 周昇亮
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市君旺印务有限公司印刷
 - ◆ 开本: 800×1000 1/16
印张: 19.5
字数: 461千字 2018年2月第1版
印数: 1-3 500册 2018年2月河北第1次印刷
- 著作权合同登记号 图字: 01-2017-9187号
-

定价: 69.00元

读者服务热线: (010)51095186转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京东工商广登字 20170147 号

版权声明

Copyright © 2017 Packt Publishing. First published in the English language under the title *Introduction to JVM Languages*.

Simplified Chinese-language edition copyright © 2018 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由Packt Publishing授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

前 言

Java虚拟机（Java Virtual Machine, JVM）是一个成熟的全能型软件运行平台，可充分利用现代硬件的功能。虽然基于Java的应用程序一度被认为速度缓慢、体态臃肿且极耗内存，但多年后的今天，情况已得到极大的改善。基于云的主流服务和网站通常要同时为数以万计的用户提供服务，它们很多都使用了基于JVM的后端，这绝非偶然。

开发运行在JVM上的应用程序时，用得最多的语言无疑是Java，但其他语言也越来越流行。本书介绍5种基于JVM的语言：Java、Scala、Clojure、Kotlin和Groovy。在这些语言中，有静态类型的，也有动态类型的；有面向对象的编程语言，也有函数式编程语言。JVM多才多艺，能够支持所有这些类型的语言。

通过在一本书中介绍这些语言，让你能够通过创建示例项目来轻松地比较它们，从而有望找出你最喜欢的语言。

涵盖的内容

第1章简要地概述Java平台和Java虚拟机（JVM）。该章描述运行在JVM上的应用程序的常见用途，包括Web应用程序、大数据分析和物联网（Internet of Things, IoT），还介绍最重要的JVM概念，如即时编译器、类型系统和垃圾收集器。

第2章从技术角度更详细地阐述JVM，包括如何在主要的操作系统（Windows、macOS和Linux）上安装Java开发包（Java Development Kit, JDK）、JDK的组织结构、Java类库的组织结构，以及如何通过设置类路径（ClassPath）来运行基于JVM的应用程序。

第3章介绍Java基础知识，包括创建类以及根据它实例化对象、在类中添加方法和属性，以及Java访问限定符和其他限定符。另外，还讨论了其他一些概念，包括抽象类、接口、数组、集合和异常。最后，介绍了线程和lambda等高级概念。

第4章详细介绍如何使用Java语言创建简单的Web服务。在创建简单Web服务的过程中，使用的工具包括Eclipse IDE、构建工具Gradle、SparkJava（一个微型Web服务框架）等编程库，以及单元测试框架JUnit。

第5章讨论既是面向对象编程语言又是函数式编程语言的Scala。该章介绍了Scala的安装过程以及它自带的交互式shell的用法；通过使用这个交互式shell，你可动态地输入并执行Scala代码，而无需先对代码进行编译。另外，还讨论了Scala的面向对象功能和函数式编程功能。

第6章详细介绍如何使用流行工具包Akka创建一个基于控制台的简单应用程序。Akka是一个专门为编写可伸缩的应用程序而设计的工具包，这种应用程序能够充分利用现代的多核处理器。该章详细讨论了很多Akka概念，如基于Actor的系统。为构建项目，使用了Scala Build Tool（Scala Build Tool，SBT），还使用了ScalaTest库来编写单元测试。

第7章介绍Clojure的基础知识。Clojure是一种动态的函数式编程语言，其设计灵感来自并非面向对象的语言Lisp。与Scala一样，Clojure也自带了一个交互式shell，可用于执行该章提供的各个示例。该章还讨论了一种用于在多线程应用程序中处理状态的技术——代理。

第8章详细介绍如何开发两个较小的项目。其中一个项目基于函数式编程语言（尤其是Lisp）中常用的技术monad，另一个项目是一个Web应用程序，它是使用流行的微型Web框架Luminus开发的。构建这两个项目时，使用的构建工具都是Leiningen。

第9章讨论JetBrain推出的静态类型编程语言Kotlin。该章阐述了Kotlin提供的可安全地处理null的类型系统，讨论了数据类、lambda和内联函数等功能，还介绍了Kotlin的过程性编程功能。

第10章详细介绍如何使用工具包JavaFX创建一个基于GUI的桌面应用程序。为构建这个项目，使用了Apache Maven；而为查找并修复bug，使用了Eclipse IDE的调试器。

第11章介绍动态编程语言Groovy。Groovy是最先推出的JVM语言之一，虽然在很大程度上它是动态语言，但也支持编译静态类型的代码，该章对这两种使用方式都做了介绍。另外，该章还探索了Groovy开发包，这是随Groovy语言一起发布的一个库，包含大量的内置类。

第12章详细介绍如何使用Groovy创建一个Web服务。这个Web服务是使用Vert.x框架创建的，它使用Java Database Connectivity（JDBC）标准从内嵌的数据库管理系统获取数据，并使用Groovy开发包中的类来生成XML。

附录A介绍了另外5种基于JVM的语言，它们大多是主流语言的方言：Oracle Nashorn（JavaScript）、Jython（Python）、JRuby（Ruby）、Frege（Haskell）和Ceylon（Red Hat推出的一种静态类型语言）。

附录B给出了各章末尾的小测验的答案。

需要什么

为最大限度地发挥本书的作用，需要一台现代的笔记本电脑或台式机，它使用最新版的Windows、macOS或Linux（最好是Ubuntu）操作系统，且至少有4GB内存（但越大越好）。

本书假定读者对使用的操作系统有一定的了解，能够熟练地安装程序以及将目录添加到环境变量中。

为谁而写

本书是为对JVM感兴趣并想深入了解最流行的JVM开发语言的程序员编写的，并假定读者使用过支持面向对象编程的现代语言，如JavaScript、Python、C#、VB.NET或C++。

排版约定

为将不同类型的信息区分开来，本书使用了很多文本样式。下面列出其中一些样式及其含义。

正文中的代码、数据库表名、用户输入，使用如下样式：“然后，我们对这个对象实例调用setName方法。”

代码块使用如下样式：

```
Product p = new Product();  
p.setName("Box of biscuits");
```

对于代码块中要突出的部分，使用粗体：

```
public String getName() {  
    return name;  
}
```

命令行输入或输出使用如下样式：

```
nano /etc/profile
```

新术语和重要词语使用黑体。



此图标表示警告或重要的注意事项。



此图标表示提示和技巧。

读者反馈

欢迎提供反馈，请将你对本书的看法告诉我们：哪些方面是你喜欢的，哪些方面你不喜欢。读者的反馈对我们来说很重要，因为这可帮助我们推出可最大限度发挥其功效的著作。

要给我们提供反馈，只需向feedback@packtpub.com发送电子邮件，并在主题中指出书名。

如果你有擅长的主题，并有志于写书或撰稿，请参阅www.packtpub.com/authors的撰稿指南。

客户支持

购买本社出版的图书后，你将获得各种帮助，让你购买的图书最大限度地发挥其效用。

下载示例代码

你可使用自己的账户从<http://www.packtpub.com>下载本书的示例代码文件。如果你是从其他地方购买的本书，可访问<http://www.packtpub.com/support>并注册，以便我们通过电子邮件将示例代码文件发送给你。

要下载代码文件，请执行如下步骤。

- (1) 登录本社网站或使用你的邮件地址注册。
- (2) 将鼠标指向网页顶端的SUPPORT选项卡。
- (3) 单击Code Downloads & Errata。
- (4) 在Search框中输入书名。
- (5) 选择要下载哪本书的示例代码文件。
- (6) 从下拉列表中选择你是从哪里购买的。
- (7) 单击Code Download。

下载文件后，使用如下软件的最新版将其解压缩：

- Windows系统请使用WinRAR或7-Zip
- Mac系统请使用Zipeg、iZip或UnRarX
- Linux系统请使用7-Zip或PeaZip

本书的示例代码也可从GitHub（<https://github.com/PacktPublishing/Introduction-to-JVM-Languages>）下载；我们提供了众多图书和视频的配套代码，你可从<https://github.com/PacktPublishing>下载。

下载彩色图片

我们还提供了一个PDF文件，其中包含了本书用到的屏幕截图和图表的彩色图片。这些彩色图片将有助于你更好地理解输出的变化，你可从<http://www.it-ebooks.info/book/1990>下载。

勘误

我们万分小心，力图让图书的内容准确无误，但即便如此，错误也在所难免。如果你在本社出版的图书中发现错误（无论是正文还是代码中的错误），请告诉我们，我们将感激不尽。通过这样做，你将让其他读者免遭同样的挫折，还可帮助我们改进该书的后续版本。无论你发现什么错误，都请告诉我们；为此你可访问<http://www.packtpub.com/submit-errata>，输入书名，单击链接Errata Submission Form，再输入你发现的错误的详情。^①你提交的勘误得到确认后，将被上传到我们的网站或添加到既有的勘误列表中。

要查看已提交的勘误，请访问<https://www.packtpub.com/books/content/support>，并在搜索框中输入书名，Errata栏将列出你搜索的信息。

打击盗版

在网上发布盗版材料是个屡禁不绝的问题。在保护版权和许可方面，本社的态度非常严肃，如果你在網上看到本社作品的非法复制品，请马上把网址或网站名告诉我们，以便我们能够采取补救措施。

请通过copyright@packtpub.com与我们联系，并提供你怀疑的盗版材料的链接。

对于你为保护我们的作者和提供有价值内容的能力提供的帮助，我们感激不尽。

问题

无论你有什么与本书相关的问题，都可通过questions@packtpub.com与我们联系，我们将竭尽全力去解决。

电子书

扫描如下二维码，即可购买本书电子版。



^① 中文版勘误可到www.it-ebooks.com.cn/book/1990查看和提交。——编者注

致 谢

感谢Packt出版社的每位员工，是他们的辛勤劳动才让本书得以付梓。感谢编辑Nitin、Vikas和Subhalaxmi以及技术审校Ramasubramanian。感谢父亲Anton和母亲Irene以及兄弟Alexander、Ruben和Wendy的大力支持！感谢我的家人和朋友。这里要特别感谢Erik、Guy、Mallory、Job、Jenna和Nina在我编写本书期间给予的支持和鼓励，还有Natalie和Marco带给我的美好回忆。最后，感谢CloudSuite的同事，尤其是Corné、Rob、Eméli和Berthold。

谨以此书献给异常风趣、善良而机灵的Melissa和Esmee Hulstein。

目 录

第 1 章 Java 虚拟机	1	第 2 章 Java 虚拟机开发	18
1.1 JVM 实现	1	2.1 JDK	18
1.2 为何要在 JVM 上开发	2	2.1.1 安装 JDK	19
1.2.1 JVM 适应市场的变化	2	2.1.2 探索 JDK	23
1.2.2 Java 类库	3	2.1.3 JRE	27
1.2.3 生态系统	3	2.2 使用包组织类	28
1.3 常见的用途	5	2.2.1 包是什么	28
1.3.1 Web 应用程序	5	2.2.2 选择包名	29
1.3.2 大数据	5	2.2.3 包名举例	30
1.3.3 IoT	6	2.2.4 全限定类名	30
1.4 JVM 概念	6	2.3 Java 类库	30
1.4.1 虚拟机	6	2.3.1 Java 类库的组织结构	31
1.4.2 JIT 编译器	7	2.3.2 包概述	31
1.4.3 基本数据类型	7	2.3.3 java.lang 包中的重要类	32
1.4.4 类	8	2.3.4 集合 API——java.util. ArrayList 和 java.util. HashMap	35
1.4.5 引用类型	8	2.4 从命令行运行 JVM 应用程序	40
1.4.6 垃圾收集器	9	2.4.1 至少有一个类包含静态方法 main()	41
1.4.7 向后兼容	11	2.4.2 存储类文件的目录结构	41
1.4.8 构建工具	11	2.4.3 为 JVM 实例设置 ClassPath	42
1.5 Java 版本	12	2.4.4 将类文件放在 JAR 归档文件 中	43
1.5.1 Java SE	12	2.4.5 使用命令 java 运行程序	44
1.5.2 Java EE	13	2.4.6 在 JVM 中运行的示例项目	46
1.5.3 Java ME	13	2.5 Eclipse IDE	49
1.6 其他 JVM 语言	14	2.5.1 下载 Eclipse IDE	50
1.6.1 为何选择其他语言	14	2.5.2 安装 Eclipse IDE	51
1.6.2 在同一个项目中使用多种 JVM 语言	15	2.6 小结	52
1.6.3 使用另一种语言编写单元测试	17		
1.7 小结	17		

第3章 Java	53	5.4 Scala 语法和规则	108
3.1 Java 中的面向对象编程功能	53	5.4.1 静态类型语言	108
3.1.1 定义类	54	5.4.2 可修改的变量和不可修改的变量	108
3.1.2 类访问限定符	54	5.4.3 常用的 Scala 类型	109
3.1.3 类限定符 final——锁定类	54	5.5 Scala 的 OOP 功能	110
3.1.4 定义包	55	5.5.1 定义包和子包	111
3.1.5 导入类	55	5.5.2 导入成员	112
3.1.6 添加类成员——变量和方法	56	5.5.3 定义类	112
3.1.7 限定符	57	5.5.4 实例变量和实例方法	113
3.1.8 构造函数和终结方法	62	5.5.5 构造函数	114
3.1.9 向上转换和向下转换	69	5.5.6 扩展类	115
3.2 编写 Java 代码	70	5.5.7 重载方法	116
3.2.1 运算符	70	5.5.8 抽象类	116
3.2.2 条件检查	71	5.5.9 特质	117
3.2.3 POJO	73	5.5.10 单例对象	118
3.2.4 数组	74	5.5.11 运算符重载	118
3.2.5 泛型和集合	75	5.5.12 Case 类	119
3.2.6 循环	77	5.6 Scala 标准库	120
3.2.7 异常	79	5.6.1 泛型	120
3.2.8 线程	81	5.6.2 集合	121
3.2.9 lambda	83	5.6.3 XML 处理	123
3.3 编程风格指南	84	5.7 Scala 的函数式编程功能	124
3.4 小测验	85	5.7.1 使用函数遍历集合	125
3.5 小结	86	5.7.2 映射-过滤-归约设计模式	125
第4章 Java 编程	87	5.7.3 柯里化	126
4.1 配置 Eclipse IDE	87	5.8 小测验	127
4.2 使用 Java 创建 Web 服务	88	5.9 小结	128
4.2.1 在 Eclipse IDE 中新建 Gradle 项目	89	第6章 Scala 编程	129
4.2.2 修改 Gradle 构建文件	90	6.1 Scala IDE for Eclipse 插件	129
4.2.3 构建项目	91	6.1.1 安装 Scala IDE for Eclipse	129
4.2.4 编写后端类	92	6.1.2 切换到 Scala IDE 透视图	131
4.3 小结	103	6.2 SBT	131
第5章 Scala	104	6.2.1 安装 SBT	132
5.1 安装 Scala	104	6.2.2 创建基于 SBT 的 Eclipse IDE 项目	132
5.2 Scala 的 REPL shell	106	6.2.3 Scala 编译器 (scalac)	135
5.3 函数式编程和命令式编程	106	6.3 创建 Akka 项目	136

6.3.1	在 SBT 构建文件中添加 Akka 依赖项.....	137	8.4.1	Eclipse IDE 中的 Clojure REPL.....	183
6.3.2	更新 Scala IDE 项目.....	138	8.4.2	更新项目的 Clojure 版本.....	183
6.3.3	Akka 概念.....	138	8.4.3	添加依赖.....	184
6.3.4	创建第一个 Akka Actor—— QuotesHandlerActor.....	140	8.5	以测试驱动开发的方式探索 monad.....	185
6.3.5	创建消息.....	142	8.6	Web 框架 Luminus.....	189
6.3.6	编写基于 ScalaTest 的单元 测试.....	144	8.6.1	创建 Luminus 项目.....	190
6.3.7	实现消息处理程序.....	146	8.6.2	将项目导入 Counterclockwise.....	191
6.3.8	创建 QuotePrinterActor.....	147	8.6.3	探索 Luminus 项目.....	191
6.3.9	主应用程序.....	149	8.6.4	在 Web 应用程序中添加页面.....	192
6.4	小结.....	151	8.7	小结.....	194
第 7 章	Clojure	152	第 9 章	Kotlin	196
7.1	安装 Clojure.....	152	9.1	安装 Kotlin.....	196
7.2	Clojure 的交互式 shell (REPL).....	154	9.2	Kotlin 的 REPL 交互式 shell.....	198
7.3	Clojure 语言.....	155	9.3	Kotlin 语言基础.....	200
7.3.1	语法.....	155	9.3.1	定义局部变量.....	200
7.3.2	表达式.....	156	9.3.2	定义函数.....	201
7.3.3	定义变量.....	157	9.3.3	Kotlin 类型.....	202
7.3.4	定义函数.....	157	9.3.4	循环.....	207
7.3.5	数据结构.....	158	9.4	Kotlin 的 OOP 功能.....	208
7.4	使用 Java 类.....	167	9.4.1	定义包.....	208
7.5	使用代理管理状态.....	169	9.4.2	导入成员.....	208
7.6	风格指南.....	172	9.4.3	定义类和构造函数.....	209
7.7	小测验.....	173	9.4.4	给类添加成员.....	210
7.8	小结.....	174	9.4.5	继承.....	212
第 8 章	Clojure 编程	175	9.4.6	接口.....	213
8.1	Eclipse IDE 插件 Counterclockwise.....	175	9.4.7	可见性限定符.....	214
8.1.1	安装插件 Counterclockwise.....	176	9.4.8	单例对象和伴生对象.....	214
8.1.2	切换到 Java 透视图.....	177	9.4.9	数据类.....	216
8.2	构建工具 Leiningen.....	177	9.4.10	lambda 和内联函数.....	217
8.3	创建可执行的 Clojure 程序.....	179	9.5	Kotlin 过程性编程.....	218
8.3.1	在不使用 Leiningen 的情况下 将代码编译成类文件.....	179	9.6	风格指南.....	219
8.3.2	使用 Leiningen 编译项目.....	180	9.7	小测验.....	220
8.4	新建 Counterclockwise 项目.....	181	9.8	小结.....	220
			第 10 章	Kotlin 编程	222
			10.1	Eclipse IDE Kotlin 插件.....	222

10.1.1	安装 Eclipse IDE Kotlin 插件	222	11.3.2	集合	257
10.1.2	切换到 Kotlin 透视图	223	11.4	动态和静态编程	260
10.2	Apache Maven	224	11.4.1	元编程	261
10.2.1	安装 Apache Maven	224	11.4.2	Groovy 静态编程	262
10.2.2	下载预制的 Kotlin 基本套件	225	11.5	小测验	264
10.2.3	在 Eclipse IDE 中导入项目	226	11.6	小结	265
10.2.4	探索构建文件 pom.xml	227	第 12 章 Groovy 编程		266
10.2.5	在 Eclipse 中更新构建文件	228	12.1	安装 Groovy Eclipse 插件	266
10.3	创建 JavaFX 桌面 GUI 应用程序	229	12.2	Apache Ivy 和 IvyDE	268
10.3.1	定制项目	230	12.3	创建并配置项目	269
10.3.2	创建可运行的应用程序	230	12.3.1	新建 Groovy Eclipse 项目	269
10.3.3	编写扩展函数	233	12.3.2	创建供 Ivy 使用的 ivy.xml 文件	270
10.3.4	布局窗格	235	12.4	Java Database Connectivity (JDBC)	272
10.3.5	实现基于 BorderPane 的布局	236	12.4.1	H2 数据库	274
10.3.6	实现动画	238	12.4.2	创建内存数据库	274
10.3.7	调试程序	241	12.5	使用 MarkupBuilder 生成 XML	278
10.4	小结	243	12.6	微服务平台 Vert.x	281
第 11 章 Groovy		244	12.6.1	在文件 ivy.xml 中添加 Vert.x 依赖	282
11.1	安装 Groovy	244	12.6.2	创建 Web 服务	283
11.2	Groovy 语言	247	12.7	小结	286
11.3	Groovy 开发包 (GDK)	255	附录 A 其他 JVM 语言		287
11.3.1	Groovy 字符串 (GString)	256	附录 B 小测验答案		296

Java虚拟机



Java虚拟机（Java Virtual Machine, JVM）是一个可用于开发和部署软件的现代平台。顾名思义，最初开发它旨在支持使用Java语言编写的应用程序，但设计Java的人不久就认识到，JVM不仅可运行Java语言，还可利用Java的功能和庞大的类库。

1995年，Sun公司^①发布了Java和第一个JVM实现。鉴于其重点是网络应用程序，Java很快就大行其道；它还被设计成可随处运行。开发Java的初衷是用于机顶盒编程，但Sun公司发现彼时机顶盒市场还不成熟，因此决定同时将这个平台推向台式机。为此，Sun公司设计了一种独特的二进制可执行格式，并称之为Java字节码。要运行被编译成Java字节码的程序，系统必须安装JVM实现。

本书将简要地介绍5种最流行的JVM语言。通过学习这些语言的基础知识，并动手编写代码，你将能够做出判断，确定哪种语言对你、你的团队和项目来说是最合适的。

我们先来说点实在的，到第2章再深入介绍Java开发包（JDK）和Java类库。当前，可使用的编程语言和平台众多，它们相互争夺市场，因此有必要先来详细地说说JVM向开发人员提供了什么。有鉴于此，本章将介绍如下主题：

- 为何要在JVM上进行开发；
- JVM的常见用途；
- JVM概念简介；
- Java版本；
- 其他JVM语言。

1.1 JVM 实现

需要指出的是，本书只考虑与Oracle Java SE 8（和更高版本）兼容的JVM实现。这个版本可

^① Sun公司于2009年4月被Oracle公司收购。——编者注

在台式机、服务器和众多单板计算机（包括尺寸如信用卡的所有Raspberry Pi）上安装。本书使用的是Oracle的JVM实现，你也可使用开源的OpenJDK和IBM的J9 Java SE实现。

本书不涵盖Google发布的用于Android手机和平板电脑的Java平台，因为用于Android的Java版本基于较旧的Java版本。虽然用于Android的Java平台版本越来越新，但它并未提供Oracle Java SE 8的所有功能，且需要使用不同的编译器和工具。另外，Google删除了大量的Java SE API，取而代之的是Google自己开发的不兼容的API。然而，本书介绍的有些语言也可用于Android开发。例如，Kotlin就是一种非常流行的Android开发语言，但本书不会对此展开讨论。

1.2 为何要在 JVM 上开发

当前，可供使用的编程语言和平台众多，为何要在JVM上开发和部署项目呢？因为JVM最初是为Java语言开发的，而近年来，其他语言的拥趸无数次地宣称，Java语言已过时乃至死亡。

近年来，流行的编程语言如走马灯似的更换，而Java犹如常青树，始终处于全球使用最多的编程语言排行榜的前列。

JVM平台为何如此强大呢？下面来看看其中一些最重要的原因：

- 它适应市场的变化，从而确保与时俱进；
- 内置的Java类库非常强大；
- 它有无可比拟的生态系统。

1.2.1 JVM 适应市场的变化

Java于20世纪90年代中期面世，那时的计算机装备的都是单核CPU，内存量也没有几个GB，因为内存条贵得不得了。Java是与时俱进的语言之一：多核CPU面世后不久，Java就通过在多个线程中运行代码来支持多核。但它并没有就此止步，而是在每次推出新版本时，都添加了让并发编程更容易的新类。这种做法到现在依然没有停止。

函数式编程范式大行其道后，Java在核心语言中新增了对lambda和流的支持。虽然Java提供这种支持的时间很晚，但相比于其他流行的语言，其实现更佳。这是因为程序员几乎什么都不需要做就能实现多线程。

适应市场变化还意味着有时需要做减法。Java面世时，热点是直接浏览器中运行Java代码。这些微型程序被称为applet，要求浏览器和系统安装专用的浏览器插件。而现在，市场已将JavaScript作为创建交互式网站的标准语言。有鉴于此，Oracle最近摒弃了applet标准。